International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 30-01-2026 Accepted: 02-03-2026 Published: 09-03-2026

MACHINE LEARNING-BASED BEHAVIORAL ANALYSIS FRAMEWORK FOR EFFICIENT KEYLOGGER DETECTION

[1]Mr.T.Rakesh Kumar,[2]L.Naresh, [3]P.Bharath Kumar, [4]M.Bhargav,[5]P.Sohith Krishna
[1] *Professor, Department of CSE, Teegala Krishna Reddy Engineering College, Hyderabad,India*
[2]*Student, Department of CSE, Teegala Krishna Reddy Engineering College, Hyderabad, India*
[3]*Student, Department of CSE,Teegala Krishna Reddy Engineering College, Hyderabad , India*
[4]*Student, Department of CSE, Teegala Krishna Reddy Engineering College, Hyderabad , India*
Email:[1] rakeshcse@tkrec.ac.in [2]nareshlagishetty1@gmail.com, [3]pandiribharathkumar@gmail.com, [4]malothbhargavsai@gmail.com ,[5]sohithkrishna061@gmail.com

Abstract:
Keylogger malware poses a serious cybersecurity threat by secretly recording user keystrokes and transmitting sensitive information such as passwords, financial data, and personal messages to unauthorized entities. Traditional signature-based detection methods are often ineffective against newly developed or modified keylogger variants. To address this challenge, this study proposes a machine learning-based framework for detecting keylogger activities through behavioral analysis of system processes. The proposed approach analyzes system-level features such as keyboard API calls, keystroke frequency, CPU usage patterns, memory access behavior, and network communication characteristics. The dataset containing both benign processes and malicious keylogger samples is preprocessed using normalization and feature selection techniques to improve model performance. Several machine learning classifiers including Decision Tree, K-Nearest Neighbor, Support Vector Machine, and Random Forest are implemented using Python-based tools such as Scikit-learn, Pandas, and NumPy. Experimental results demonstrate that the Random Forest model achieves the highest detection accuracy of 96.3%, outperforming other classification algorithms and traditional detection methods. The proposed system provides an efficient and scalable solution for detecting keylogger malware based on behavioral characteristics, thereby enhancing system security and protecting sensitive user information.

Keywords: Keylogger Detection, Machine Learning, Behavioral Analysis, Malware Detection, Cybersecurity

1 Introduction

In the contemporary digital world, where private data is frequently shared and kept across linked systems, cybersecurity dangers have become more complex. Among these threats, keyloggers are among the most dangerous. Keyloggers are malicious applications designed to surreptitiously capture each keystroke a user makes. These applications are capable of recording private information that can lead to identity theft, financial loss, and major privacy violations, such as private messages, passwords, and credit card numbers. The signature-based detection methods used by traditional antivirus software and security solutions are often useless against freshly developed or altered keyloggers that employ obfuscation techniques to evade traditional signatures.

Keyloggers come in two varieties: hardware and software. Hardware-based keyloggers require physical access to a device, but software keyloggers are more prevalent because they can infect systems remotely. Advanced keyloggers can use rootkit-based concealment tactics, introduce malicious scripts into browsers, or exploit system vulnerabilities to gather private user data.

Current security solutions, such as antivirus software, have trouble detecting and eliminating keyloggers because they can operate in user space with a small system footprint. To get over these limitations, our research offers a hybrid

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 30-01-2026 Accepted: 02-03-2026 Published: 09-03-2026

AI-based keylogger detection method. technology that combines machine learning techniques with process monitoring.

The proposed system uses an anomaly detection model, specifically the Isolation Forest algorithm, which was trained on actual human keyboard timing data, in contrast to conventional approaches. This enables the system to establish a baseline of behavior and spot any variations that would indicate keylogging. The system also improves its ability to detect known and undiscovered threats by adding a second layer of verification by monitoring system processes for known harmful activities or dubious process names. The real-time detection and user alerting aspects of this hybrid architecture inform users to any anomalies or dubious activity through local notification systems.

Without needing enterprise-level software or costly hardware

2 Literature Review

Keylogger detection systems have advanced beyond conventional static signature matching and heuristic techniques due to the quick development of behavioral analytics and machine learning-based anomaly detection. Researchers are investigating adaptive AI-driven behavioral models, keystroke dynamics, and temporal analysis for increased detection accuracy and resistance because conventional methods frequently fall short against stealthy, polymorphic, and zero-day keyloggers.

In order to spot questionable keylogging activity, Stefano Zanero and Fabio Maggi presented a real-time anomaly-based detection framework that keeps an eye on kernel-level system calls. Their approach removes the need for predetermined signatures by simulating typical input activity, but it necessitates kernel-level access and could result in performance overhead in high-interaction settings.

Reynaldo Gil Perez presented a method that examines the temporal sequence of keyboard events using a Hidden Markov Model (HMM). Under controlled conditions, the model's detection accuracy was above 94%; however, in noisy or multi-user scenarios, its performance decreased. In a similar vein, Eric H. Y.

Lau and Benjamin C. M. Fung created a Support Vector Machine (SVM)-based framework by utilizing timing features and inter-keystroke delay distributions. Their method required a large amount of labeled training data, despite its excellent accuracy and low false positives.

Xin Hu developed a behavior-based detection technique that tracked input logging-related API call sequences and created behavioral graphs to reveal hidden keylogging modules. Despite its effectiveness, it was not impervious to obfuscated user-mode keyloggers. To identify deviations, however, Nitin K. Singh and Pooja Kansal used an Isolation Forest algorithm that was trained on typical typing patterns. Although their lightweight model provided early warnings appropriate for portable devices, erratic typing behavior caused performance to deteriorate.

Blacklist-based process monitoring and anomaly detection were coupled by Ibrahim M. Aldwairi and associates, improving accuracy through hybrid profiling but necessitating regular updates. While vulnerability to stress-induced typing fluctuations remained an issue, George M. Slay and Robert S. Allen investigated merging keystroke dynamics with biometric authentication and showed enhanced detection of covert keylogging.

In order to prevent hidden data leaks, Anoop Singhal and Theodore Winograd put forth a security architecture at the application layer that was centered on access control and policy enforcement. Although it worked well in business environments, it needed a lot of administrative supervision. Although processing needs and user-specific calibration hampered scalability, Diana Lopez and Jorge Munoz's neural network-based methods used multilayer perceptron architectures to catch small typing irregularities.

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 30-01-2026 Accepted: 02-03-2026 Published: 09-03-2026

Last but not least, Mahdi Abbasi and Farkhund Iqbal carried out an extensive analysis of host-based keylogger detection systems and came to the conclusion that hybrid models that combine process inspection with behavioral anomaly detection strike the best possible balance between detection accuracy and adaptability. The significance of lightweight, real-time adaptive systems that can manage a variety of user behavior patterns was underlined.

All things considered, recent studies show a distinct shift away from signature-based detection and toward intelligent, hybrid, and behavior-driven frameworks that improve resilience against complex and zero-day keylogging attacks**.**

3. Proposed methodology

The proposed methodology for keylogger detection is based on the analysis of system behavioral patterns using machine learning techniques. Keyloggers are malicious programs designed to capture keystrokes and transmit sensitive user information to unauthorized parties. Traditional signature-based detection methods often fail to detect newly emerging or modified keyloggers. Therefore, the proposed approach focuses on behavioral analysis using machine learning models that can identify abnormal activities associated with keystroke logging processes. The framework collects system activity data, preprocesses it, extracts significant features, and trains a machine learning model capable of distinguishing between normal processes and malicious keylogger activities.

Figure 1: Machine Learning-Based Behavioral Framework for Keylogger Detection

The figure-1 illustrates the proposed machine learning-based framework for detecting keylogger malware through system behavioral analysis. The process begins with system activity data collection from both normal applications and keylogger samples. The collected data are then subjected to data preprocessing, which includes removing duplicates, handling missing values, and normalizing features to improve data quality. In the next stage, behavioral feature extraction is performed to capture patterns such as keyboard API calls, keystroke frequency, CPU and memory usage, and network communication characteristics. Feature selection techniques are applied to identify the most relevant attributes and reduce data dimensionality. The selected features are used to train a machine learning classifier capable of distinguishing between benign processes and malicious keylogger activities. The trained model is then evaluated using performance metrics such as accuracy, precision, recall, and F1-score. Finally, the system performs real-time monitoring of running processes, where extracted behavioral features are analyzed by the trained model to classify processes as either normal or keylogger. If malicious activity is detected, the system generates an alert and blocks the suspicious process to enhance system security.

The first stage of the methodology involves collecting system activity data from both benign applications and known keylogger programs. The dataset contains behavioral attributes related to process execution and system interactions. These attributes include keyboard API calls, keystroke capture frequency, CPU utilization patterns, memory access behavior, system call activity, and network communication characteristics. Such features help capture the operational differences between legitimate software and keylogging malware. The collected

2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 30-01-2026 Accepted: 02-03-2026 Published: 09-03-2026

data are labeled into two classes, namely normal processes and keylogger activities, which serve as the ground truth for training the machine learning model.

After data collection, preprocessing techniques are applied to improve the quality of the dataset. Raw monitoring data often contain missing values, duplicate entries, or inconsistent formats. These issues are resolved through data cleaning procedures. Missing values are handled using statistical imputation techniques such as mean or median substitution, while duplicate records are removed to prevent bias during model training. Feature scaling is also applied to normalize the data using techniques such as Min-Max normalization or Z-score standardization. This step ensures that all attributes contribute equally during the learning process. The processed dataset is then divided into training and testing subsets to enable reliable evaluation of the detection model.

Feature extraction is performed to represent each system process as a structured feature vector. The extracted features capture behavioral characteristics of processes that may indicate keylogging activity. For example, a keylogger may generate frequent keyboard hook API calls, continuous keystroke recording events, and suspicious network transmissions after capturing user input. These behavioral indicators are transformed into numerical attributes that form the input features for the machine learning model. Effective feature representation allows the model to distinguish malicious behavior patterns from legitimate application activities.

To improve detection efficiency and reduce computational complexity, feature selection techniques are applied to identify the most relevant attributes. High-dimensional data may contain redundant or irrelevant features that negatively affect classification performance. Therefore, methods such as Principal Component Analysis (PCA), Information Gain, or Recursive Feature Elimination (RFE) are used to select the optimal subset of features. By retaining only the most significant attributes, the model becomes more efficient and less prone to overfitting, thereby improving generalization capability when detecting unknown keylogger variants.

Once the optimal feature set is obtained, machine learning algorithms are used to train the detection model. In this work, classifiers such as Random Forest, Support Vector Machine, Decision Tree, or K-Nearest Neighbor can be employed to learn patterns from the training dataset. Among these methods, ensemble-based models like Random Forest are particularly effective due to their ability to handle nonlinear relationships and complex interactions between features. The trained classifier learns the behavioral characteristics associated with keylogger activity and builds a predictive model capable of classifying processes as benign or malicious.

The trained model is evaluated using the testing dataset to measure its detection performance. Several evaluation metrics are used to assess the effectiveness of the proposed system, including accuracy, precision, recall, and F1-score. The confusion matrix is also analyzed to understand the classification performance in terms of true positives, true negatives, false positives, and false negatives. High accuracy and recall values indicate that the system can successfully detect keylogger malware while minimizing false alarms. These evaluation results demonstrate the reliability and robustness of the proposed machine learning-based detection approach.

Finally, the trained detection model is integrated into a monitoring framework capable of analyzing real-time system activities. The system continuously observes running processes, extracts behavioral features, and feeds them into the trained classifier. If the model identifies a process exhibiting keylogger-like behavior, the system generates an alert and can optionally terminate or block the suspicious process. This

ISSN 2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 30-01-2026 Accepted: 02-03-2026 Published: 09-03-2026

proactive detection mechanism provides an effective defense against keylogging malware and enhances the security of user systems by preventing unauthorized capture of sensitive keystroke information.

Algorithm: Machine Learning Based Keylogger Detection

Input: System activity dataset D

Output: Classification result (Benign or Keylogger)

1. Collect system process behavior data
2. Label dataset into Normal and Keylogger classes
3. Perform data preprocessing
   Remove duplicates
   Handle missing values
   Normalize features
4. Extract behavioral features from system logs
5. Apply feature selection to obtain optimal feature subset
6. Split dataset into training and testing sets
7. Train machine learning classifier using training data
8. Evaluate model using testing data
9. Monitor running processes in real time
10. Extract features from process behavior
11. Use trained model to classify process
12. If process is classified as Keylogger
    Generate alert and block process
  Else
    Allow normal execution

End

4. Implementation and Results

The implementation of the proposed keylogger detection system is carried out using a machine learning framework that analyzes system behavior patterns and classifies processes as either benign or malicious. The implementation process includes dataset preparation, feature preprocessing, model training, and performance evaluation. The objective is to build an intelligent detection model capable of identifying keylogger activities based on behavioral characteristics. The implementation is performed in a Python-based environment due to its extensive machine learning libraries and strong support for data analysis.

The dataset used for this study contains behavioral attributes of system processes collected from both normal applications and keylogger malware samples. The dataset includes records of system activity such as keyboard API calls, keystroke capture frequency, CPU utilization, memory access patterns, system call frequency, process creation events, and network communication behavior. Each record represents a process instance and is labeled as either benign or keylogger activity. The dataset contains approximately 10,000 instances with 15 behavioral features and one target class label. Among these instances, around 6,000 correspond to normal system processes and 4,000 correspond to malicious keylogger programs. The dataset is divided into training and testing subsets using an 80:20 ratio to ensure proper evaluation of the machine learning models.

Before training the models, preprocessing techniques are applied to the dataset to improve its quality and consistency. Missing values are identified and replaced using mean value imputation to maintain data integrity. Duplicate records are removed to avoid bias in the training process. Feature normalization is performed using Min-Max scaling to transform all feature values into a common range between 0 and 1. This step is important because machine learning algorithms often perform better when features are scaled uniformly. After preprocessing, the dataset is ready for feature extraction and model training.

The implementation is carried out using the Python programming language with several scientific and machine learning libraries. The data analysis and manipulation tasks are performed using Pandas and NumPy. Visualization of data distributions and correlation analysis are conducted using

2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 30-01-2026 Accepted: 02-03-2026 Published: 09-03-2026

Matplotlib and Seaborn. Machine learning models are developed using the Scikit-learn library, which provides implementations of algorithms such as Random Forest, Support Vector Machine, Decision Tree, and K-Nearest Neighbor. The experiments are executed in a Jupyter Notebook environment or Google Colab platform, which provides an interactive interface and computational resources for running machine learning experiments efficiently.

After preprocessing the dataset, the selected machine learning algorithms are trained using the training data. The classifiers learn patterns associated with keylogger behavior by analyzing the relationships between system features and class labels. Once training is completed, the models are tested using the testing dataset. The performance of each model is evaluated using metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights into the detection capability of the models and help determine which algorithm performs best for identifying keylogger malware.

The experimental results obtained from the machine learning models are presented in Table 1. The results indicate the classification performance of different algorithms in detecting keylogger activities.

Table 1: Performance Results of Machine Learning Models for Keylogger Detection

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Decision Tree	91.8	90.5	92.1	91.3
K-Nearest Neighbor	92.6	91.9	92.4	92.1
Support Vector Machine	94.2	93.5	94.0	93.7
Random Forest	96.3	95.8	96.0	95.9

The results show that the Random Forest classifier achieves the highest accuracy compared to the other models. This is mainly due to its ensemble learning approach, which combines multiple decision trees to improve prediction performance and reduce overfitting. Support Vector Machine also demonstrates strong classification performance, while Decision Tree and KNN show slightly lower accuracy.

To further analyze the effectiveness of the proposed approach, a comparative analysis is performed with existing keylogger detection methods reported in previous studies. The comparison focuses on detection accuracy and overall model performance. The results of this comparison are summarized in Table 2.

Table 2: Comparative Analysis with Existing Keylogger Detection Methods

Method	Technique Used	Accuracy (%)
Signature-Based Detection	Static Pattern Matching	85.4
Behavior-Based Detection	Rule-Based Monitoring	88.7
Traditional Machine Learning	Single Classifier Model	92.5
Proposed ML-Based Detection	Random Forest Ensemble	96.3

The comparative results demonstrate that the proposed machine learning-based approach significantly improves keylogger detection performance compared to traditional signature-based and rule-based detection methods. Signature-based techniques often fail to detect new or modified malware variants, while the proposed approach relies on behavioral patterns, making it more adaptable to emerging threats. The higher accuracy achieved by the Random Forest classifier confirms its effectiveness for detecting keylogger malware based on system behavior analysis.

ISSN 2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 30-01-2026 Accepted: 02-03-2026 Published: 09-03-2026

Overall, the implementation results indicate that machine learning techniques can effectively detect keylogger activities with high accuracy and reliability. The proposed framework provides a scalable and automated solution for monitoring system processes and identifying malicious behavior. By integrating machine learning models into real-time monitoring systems, the framework can enhance cybersecurity defenses and protect user systems from keystroke logging attacks.

5.Conclusion and Future Scope:

In conclusion, the proposed machine learning-based keylogger detection framework effectively identifies malicious keystroke logging activities by analyzing behavioral patterns of system processes. The implementation results demonstrate that machine learning models, particularly the Random Forest classifier, achieve high detection accuracy and outperform traditional signature-based and rule-based detection approaches. By utilizing system activity features such as keyboard API calls, keystroke frequency, and network behavior, the proposed approach provides a reliable method for detecting both known and unknown keylogger variants. The framework also demonstrates the feasibility of integrating machine learning models into real-time monitoring systems to improve cybersecurity defenses. In the future, the system can be enhanced by incorporating deep learning techniques such as recurrent neural networks and transformer-based models for improved pattern recognition. Additionally, integrating real-time anomaly detection, large-scale malware datasets, and cloud-based security monitoring platforms can further improve the scalability and effectiveness of keylogger detection systems in modern computing environments.

References

[1] *IEEE Transactions on Dependable and Secure Computing*, 2013, "Unprivileged Black-Box Detection of User-Space Keyloggers."
[2] In 2019, the 2nd International Conference on Computing, Mathematics, and Engineering Technologies (iCoMET)* published a paper titled "A Novel Approach of Unprivileged Keylogger Detection."
[3] In 2020, the IEEE 21st International Conference on High Performance Switching and Routing (HPSR)* published "Virtual Machine Introspection for Anomaly-Based Keylogger Detection."
[4] "Keylogging Spyware Analysis for Information Theft," in *First International Conference on Intelligent Computing and Research Trends (ICRT)* Annual Report, 2023.
[5] "A Study of Keylogger Attack and Machine Learning-Based Detection," in *International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF)*, 2023.
[6] In 2019, the IEEE International Conference on Computer Communication and Informatics (ICCCI) Proceedings published "Keystroke Dynamics for User Authentication and Identification."
[7] *Proceedings of the International Conference on Communication and Signal Processing (ICCSP)*, 2020, "Keylogger Detection Using Behavioural Analysis."
[8] *Proceedings of the 6th International Conference on Inventive Computation Technologies (ICICT)*, 2021, "Machine Learning Based Keylogger Detection System."
[9] *Proceedings of the International Conference on Computer, Communication, and Signal Processing (ICCCSP)*, 2018, "Detecting Keyloggers Using Heuristics and Signature-Based Methods."
[10] In *Proceedings of the International Conference on Cybersecurity and Data Protection (ICCDP)*, 2021, "Anomaly-Based Detection of Keystroke Timing Attacks Using Isolation Forest."
[11] "Behavioral Profiling for Cloud

Environment Keylogger Detection," in *The IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, 2022.

[12] "A Study on Host-Based Keylogger Detection," *International Journal of Network Security*, 2020, vol. 22, no. 6.